TÉCNICO LISBOA

Deep Learning

MEEC, MEIC, MECD

Final Exam

January 17, 2024

Version A

Instructions

- You have 120 minutes to complete the exam.
- Make sure that your test has a total of 13 pages, then write your full name and student number on this page (and your number in all the other pages).
- The test has a total of 18 questions, with a maximum score of 100 points. The questions have different levels of difficulty. The point value of each question is provided next to the question number.
- Please provide your answer in the space below each question. If you make a mess, clearly indicate your answer.
- The exam is open book and open notes. You may use a calculator, but any other type of electronic or communication equipment is not allowed.
- Good luck.

Part 1	**Part 2**	**Part 3, Pr. 1**	**Part 3, Pr. 2**	**Total**
32 points	18 points	25 points	25 points	100 points

Part 1: Multiple Choice Questions (32 points)

In each of the following questions, indicate your answer by choosing **one single** option. Read all the options carefully before choosing.

1. (4 points) Consider the softmax transformation and, for some vector $\boldsymbol{v} \in \mathbb{R}^K$ ($\boldsymbol{v} \neq \mathbf{0}$), the function $g(s,t) = \text{softmax}(\boldsymbol{v}\, s + \mathbf{1}_K\, t)$, where $s,t \in \mathbb{R}$ are scalars and $\mathbf{1}_K = [1,\ldots,1]^\top$ is a vector with K ones. Then,

   - ☐ $g(s,t)$ is constant with respect to both s and t.
   - ■ **$g(s,t)$ is constant with respect to t, but not with respect to s.**
   - ☐ $g(s,t)$ is constant with respect to s, but not with respect to t.
   - ☐ $g(s,t)$ is not constant with respect to s or t.

   Solution: A global shift to all the arguments of softmax does not change the output, as it corresponds to multiplying all numerators and the common denominator by the same constant. Obviously, scaling the arguments (changing the *temperature*) changes the output.

2. (4 points) Let $\mathcal{D} = \{(\boldsymbol{x}^{(1)}, y^{(1)}), \ldots, (\boldsymbol{x}^{(n)}, y^{(n)})\}$ be a linearly separable dataset for binary classification and $\mathcal{D}' = \left\{\left(\text{ReLU}(\boldsymbol{x}^{(1)}+\beta\mathbf{1}), y^{(1)}\right), \ldots, \left(\text{ReLU}(\boldsymbol{x}^{(n)}+\beta\mathbf{1}), y^{(n)}\right)\right\}$ another dataset obtained by computing the ReLU of the component of each point in $\mathcal{D}$, shifted by β. Then,

   - ☐ $\mathcal{D}'$ is also linearly separable, independently of β.
   - ☐ $\mathcal{D}'$ is not linearly separable, independently of β.
   - ■ **$\mathcal{D}'$ may or not be linearly separable, depending on the choice of β.**
   - ☐ There are datasets $\mathcal{D}$ for which $\mathcal{D}'$ cannot be linearly separable, for any β.

   Solution: Begin by noting two things: if all the points (that is, $\boldsymbol{x}^{(1)}$, ..., $\boldsymbol{x}^{(n)}$) are in the first orthant (all components are non-negative), then the ReLU does change them; shifting all the points equally does not affect the linear separability of the dataset. Now, for any dataset, it is possible to find β such that all the shifted points $\boldsymbol{x}^{(1)} + \beta\mathbf{1}$, ..., $\boldsymbol{x}^{(n)} + \beta\mathbf{1}$ are in the first orthant, thus the ReLU has no effect and the dataset remains linearly separable.

3. (4 points) In residual networks (ResNets), what is the main purpose of using skip connections?

   - ☐ To connect the input layer directly to the output layer, simplifying the learning task.
   - ☐ To skip unnecessary layers during training to reduce computational complexity.
   - ☐ To randomly drop layers during training to prevent overfitting, similar to dropout.
   - ■ **To allow the gradient to flow directly through the network, mitigating the vanishing gradient problem.**

   Solution: As shown in slide 33 of Lecture 7.

4. (4 points) Assume a transformer with two attention heads, one with projection matrices $\boldsymbol{W}_Q^{(1)}$, $\boldsymbol{W}_K^{(1)}$, $\boldsymbol{W}_V^{(1)}$ and another with projection matrices $\boldsymbol{W}_Q^{(2)}$, $\boldsymbol{W}_K^{(2)}$, $\boldsymbol{W}_V^{(2)}$ where $\boldsymbol{W}_Q^{(2)} = 2\boldsymbol{W}_Q^{(1)}$, $\boldsymbol{W}_K^{(2)} = \boldsymbol{W}_K^{(1)}$, and $\boldsymbol{W}_V^{(2)} = (1/2)\,\boldsymbol{W}_V^{(1)}$. Let $\boldsymbol{P}^{(1)}$ and $\boldsymbol{P}^{(2)}$ be the attention probability matrices and $\boldsymbol{Z}^{(1)}$ and $\boldsymbol{Z}^{(2)}$ be the context representations obtained by the two attention heads. What is the relationship between $\boldsymbol{P}^{(1)}$ and $\boldsymbol{P}^{(2)}$ and between $\boldsymbol{Z}^{(1)}$ and $\boldsymbol{Z}^{(2)}$ that is true in general?

- ☐ $\boldsymbol{P}^{(1)} = \boldsymbol{P}^{(2)}$ and $\boldsymbol{Z}^{(1)} = \boldsymbol{Z}^{(2)}$.
- ☐ $\boldsymbol{P}^{(1)} \neq \boldsymbol{P}^{(2)}$ and $\boldsymbol{Z}^{(1)} = \boldsymbol{Z}^{(2)}$.
- ☐ $\boldsymbol{P}^{(1)} = \boldsymbol{P}^{(2)}$ and $\boldsymbol{Z}^{(1)} \neq \boldsymbol{Z}^{(2)}$.
- ■ **$\boldsymbol{P}^{(1)} \neq \boldsymbol{P}^{(2)}$ and $\boldsymbol{Z}^{(1)} \neq \boldsymbol{Z}^{(2)}$.**

Solution: We have $\boldsymbol{P}^{(2)} = \text{Softmax}(\boldsymbol{Q}^{(2)}(\boldsymbol{K}^{(2)})^\top/\sqrt{K}) = \text{Softmax}(2\boldsymbol{Q}^{(1)}(\boldsymbol{K}^{(1)})^\top/\sqrt{K}) \neq \text{Softmax}(\boldsymbol{Q}^{(1)}(\boldsymbol{K}^{(1)})^\top/\sqrt{K}) = \boldsymbol{P}^{(1)}$. Consequently, $\boldsymbol{Z}^{(2)} = \boldsymbol{P}^{(2)}\boldsymbol{V}^{(2)} \neq \boldsymbol{P}^{(1)}\boldsymbol{V}^{(1)} = \boldsymbol{Z}^{(1)}$.

5. (4 points) Consider a neural network layer with preactivation $\boldsymbol{z}^{(\ell)} = \mathbf{W}\boldsymbol{h}^{(\ell-1)} + \boldsymbol{b}$ and activation $\boldsymbol{h}^{(\ell)} = \boldsymbol{g}(\boldsymbol{z}^{(\ell)})$ where the activation function is the so-called ELU (*exponential linear unit*) $g(z) = z$, if $z \geq 0$, and $g(z) = \exp(z) - 1$, if $z < 0$ (applied element-wise). Let L be the loss function associated to this neural network. Which of these expressions is the correct gradient of L with respect to $\boldsymbol{h}^{(\ell-1)}$?

   - ☐ $\frac{\partial L}{\partial \boldsymbol{h}^{(\ell-1)}} = \boldsymbol{W}^\top \left(\frac{\partial L}{\partial \boldsymbol{h}^{(\ell)}} \odot 1(\boldsymbol{z}^{(\ell)} < \mathbf{0}) \odot \exp(\boldsymbol{z}^{(\ell)})\right)$.
   - ☐ $\frac{\partial L}{\partial \boldsymbol{h}^{(\ell-1)}} = \boldsymbol{W}^\top \left(\frac{\partial L}{\partial \boldsymbol{h}^{(\ell)}} \odot 1(\boldsymbol{z}^{(\ell)} \geq \mathbf{0}) \odot \exp(\boldsymbol{z}^{(\ell)}\right)$.
   - ■ $\frac{\partial L}{\partial \boldsymbol{h}^{(\ell-1)}} = \boldsymbol{W}^\top \left(\frac{\partial L}{\partial \boldsymbol{h}^{(\ell)}} \odot \min\left\{\exp(\boldsymbol{z}^{(\ell)}), 1\right\}\right)$.
   - ☐ $\frac{\partial L}{\partial \boldsymbol{h}^{(\ell-1)}} = \boldsymbol{W}^\top \left(\frac{\partial L}{\partial \boldsymbol{h}^{(\ell)}} \odot \max\left\{\exp(\boldsymbol{z}^{(\ell)}), 1\right\}\right)$.

   Solution: We have $\frac{\partial L}{\partial z_i^{(\ell)}} = \frac{\partial L}{\partial h_i^{(\ell)}} g'(z_i^{(\ell)})$, i.e., $\frac{\partial L}{\partial \boldsymbol{z}^{(\ell)}} = \frac{\partial L}{\partial \boldsymbol{h}^{(\ell)}} \odot \boldsymbol{g}'(\boldsymbol{z}^{(\ell)})$ and $\frac{\partial L}{\partial h_i^{(\ell-1)}} = \sum_j \frac{\partial L}{\partial z_j^{(\ell)}} W_{ji}$, i.e., $\frac{\partial L}{\partial \boldsymbol{h}^{(\ell-1)}} = W^\top \frac{\partial L}{\partial \boldsymbol{z}^{(\ell)}}$. We have $g'(z_i) = 1$, if $z_i \geq 0$, and $g'(z_i) = \exp(z_i)$, if $z_i < 0$. Because, for $z_i < 0$, $\exp(z_i) < 1$, and for $z_i \geq 0$, $\exp(z_i) \geq 1$, this can be written as $g'(z_i) = \min\{\exp(z_i), 1\}$.

6. (4 points) Which of the following sentences is **false**?

   - ☐ Deep networks may have more parameters than the number of training samples and still generalize well.
   - ☐ If the training set for a binary classification problem is linearly separable, the perceptron algorithm is guaranteed to find a separation hyperplane after a finite number of steps.
   - ■ **If the training set for a binary classification problem is linearly separable, gradient descent for binary logistic regression is guaranteed to find a separation hyperplane after a finite number of steps.**
   - ☐ Convolutional layers have fewer parameters than fully connected layers with the same input and output dimensions.

   Solution: Gradient descent (stochastic or batch) may converge or not, depending on the choice of step size, and its convergence, even with an adequately chosen step size, has no guarantees of providing a separating hyperplane in a finite number of iterations.

7. (4 points) What distinguishes *variational autoencoders* (VAE) from traditional autoencoders?

   - ☐ VAEs use a different activation function in the hidden layers to enhance non-linear feature extraction.
   - ☐ VAEs encode input data into fixed points in the latent space, while traditional autoencoders use variable points.
   - ■ **VAEs introduce randomness in the encoding process, producing a distribution over the latent space for each input.**
   - ☐ VAEs solely focus on reducing the dimensionality of data, unlike traditional autoencoders which also focus on data generation.

Solution: VAEs introduce randomness in the encoding process, producing a distribution over the latent space for each input.

8. (4 points) Which of the following statements about masked attention in transformer models is **false**?

- ☐ Masked attention in the decoder allows each token to attend only to previously generated tokens, preserving the autoregressive property in sequence generation tasks.
- ■ **Masked attention mechanisms are used in the encoder to prevent each token from attending to subsequent tokens in the input sequence.**
- ☐ In masked attention, future tokens in the sequence are masked (or hidden) during training to prevent the model from using them in predictions.
- ☐ The use of masked attention is crucial for tasks like language modeling, where the prediction of a word depends on the preceding words.

Solution: Masked attention mechanisms are used in the encoder to prevent each token from attending to subsequent tokens in the output, not input, sequence.

Part 2: Short Answer Questions (18 points)

Please provide **brief** answers (1-2 sentences) to the following questions.

1. (6 points) What is the inductive bias associated to the use of a convolutional layer as the input a deep network?

   Solution: By applying the same filter across the entire input, a convolutional layer detects the same features regardless of position, making them translation equivariant. Also, they assume that small, localized regions of the input are relevant for feature etection, leading to the use of local receptive fields.

2. (6 points) When training a deep network by stochastic or batch gradient descent, with back-propagation, why is it bad idea to initialize all weights and biases at zero?

   Solution: If all weights are initialized to zero, all neurons in all layers will receive the same gradient and thus update their weights in the same way. This phenomenon is often referred to as the "symmetry problem," essentially implying that all neurons in a layer will learn the same thing, negating the benefits of having multiple neurons in a layer.

3. (6 points) In an attention mechanism, explain the role of the softmax transformation and why a hyperbolic tangent would not be an adequate choice.

   Solution: The role of the softmax transformation is to obtain a vector of weights that are non-negative and sum to one, which are then used to obtain a weighted average of the value vectors. Using a hyperbolic tangent leads to numbers that may be negative and do not sum to one, thus are not adequate weights to compute a weighted average.

Part 3: Problems (50 points)

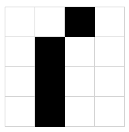
Problem 1: Convolutional Neural Networks (25 points)

Consider a convolutional layer with a 3×3 filter and a ReLU nonlinearity, where stride = 1 and padding = 1. Suppose that, after training, the parameters of the filter are

$$\boldsymbol{K} = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, \qquad\qquad b = 0,$$

where b is the bias term.

1. (7 points) Suppose that the layer receives, as input, the image



which can be represented as

$$\boldsymbol{x} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix},$$

Compute the output $\boldsymbol{h}$ of the convolutional layer when the input is x. Indicate all relevant computations.

Solution: To compute $\boldsymbol{h}$ we first add a padding of 1, yielding

$$\boldsymbol{x}_{\text{padded}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

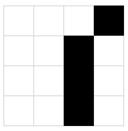
We now compute the convolution with $\boldsymbol{K}$, to get

$$\boldsymbol{z} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} -1 & -2 & 1 & 2 \\ -3 & -1 & 3 & 1 \\ -4 & 0 & 4 & 0 \\ -3 & 0 & 3 & 0 \end{bmatrix}.$$

Finally, applying the ReLU, we get

$$\boldsymbol{h} = \text{ReLU}(\boldsymbol{z}) = \begin{bmatrix} 0 & 0 & 1 & 2 \\ 0 & 0 & 3 & 1 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 3 & 0 \end{bmatrix}.$$

2. (10 points) Repeat the computation in 1, but now using the translated image



as input. Compare the output of the convolutional layer with that in 1 and identify the property that can be observed.

Solution: The new padded image is

$$\boldsymbol{x}_{\text{padded}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

The result of the convolution is now:

$$\boldsymbol{z} = \begin{bmatrix} 0 & -1 & -2 & 1 \\ 0 & -3 & -1 & 3 \\ 0 & -4 & 0 & 4 \\ 0 & -3 & 0 & 3 \end{bmatrix}.$$

Applying the ReLU, we get

$$\boldsymbol{h} = \text{ReLU}(\boldsymbol{z}) = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 3 \end{bmatrix}.$$

We can observe that a translation in the input leads to an equivalent translation in the output, a property of convolutional layers known as *equivariance*.

3. (3 points) Suppose now that the convolutional layer is followed by a 2×2 max-pool layer, with a stride of 2 and no padding. Compute the output of the max-pool layer when the input to the convolutional layer is the image in 1.

Note: If you did not solve 1, use as the output of the convolutional layer the matrix

$$\boldsymbol{h} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 3 & 0 \end{bmatrix}.$$

Solution: Applying max-pooling to the image in 1 after going through the convolutional layer yields

$$\boldsymbol{z}'_{(\text{a})} = \begin{bmatrix} 0 & 3 \\ 0 & 4 \end{bmatrix}.$$

4. (5 points) Repeat the computation in 3 when the input to the convolutional layer is now the image in 2. Compare the output of the max-pool layer with that in 3 and identify the property that can be observed.

   Note: If you did not solve 2, use as the output of the convolutional layer the matrix

$$\boldsymbol{h} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 3 \end{bmatrix}.$$

   Solution: Applying max-pooling to the image in 1 after going through the convolutional layer yields

$$\boldsymbol{z}'_{(\mathrm{b})} = \begin{bmatrix} 0 & 3 \\ 0 & 4 \end{bmatrix}.$$

   We can observe that, although the input is translated, the output is the same. This is a property known as *invariance.*

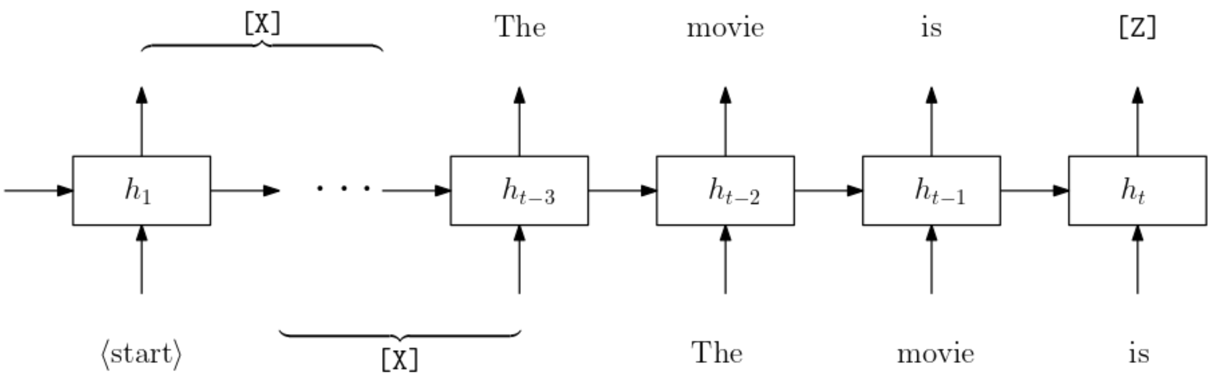
Problem 2: Prompted Generation for Sentiment Classification (25 points)

Consider a sentiment classification problem where the goal is to predict if a movie review is positive (+) or negative (−). The following prompt template is used:

`[X]` The movie is `[Z]`

where the slot `[X]` is filled with the text of the review and `[Z]` is a slot from which the sentiment prediction can be generated. Let t be the time-step corresponding to the slot `[Z]`, and let $w_1, ..., w_t$ the words of the sequence after filling the prompt template above. The sentiment prediction $\hat{y} \in \{+1, -1\}$ is computed according to the rule:

$$\hat{y} = \begin{cases} +1 & \text{if } P(w_t = \text{“good”} \mid w_1, ..., w_{t-1}) \geq P(w_t = \text{“bad”} \mid w_1, ..., w_{t-1}), \\ -1 & \text{otherwise.} \end{cases}$$



1. (10 points) Suppose that the movie review is “Awful plot and actors.” and that we use an autoregressive RNN model as the language model. The word embeddings are

$$\boldsymbol{x}_{\langle\text{START}\rangle} = [0,0,0]^\top, \quad \boldsymbol{x}_{\text{Awful}} = [0,-5,0]^\top, \quad \boldsymbol{x}_{\text{plot}} = [1,0,0]^\top, \quad \boldsymbol{x}_{\text{and}} = [0,0,0]^\top,$$
$$\boldsymbol{x}_{\text{actors.}} = [-1,0,0]^\top, \quad \boldsymbol{x}_{\text{The}} = [0,0,0]^\top, \quad \boldsymbol{x}_{\text{movie}} = [-2,0,0]^\top, \quad \boldsymbol{x}_{\text{is}} = [0,0,0]^\top,$$
$$\boldsymbol{x}_{\text{good}} = [1,2,3]^\top, \quad \boldsymbol{x}_{\text{bad}} = [-1,-2,-3]^\top,$$

and the parameters of the RNN are

$$\mathbf{W}_{hx} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & -1 & -1 \end{bmatrix}, \quad \mathbf{W}_{hh} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{W}_{yh} = \begin{bmatrix} 0 & 0 \\ 1 & 2 \\ 3 & 4 \\ 5 & 6 \\ 7 & 8 \\ 9 & 10 \\ 11 & 12 \\ 13 & 14 \\ 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{array}{l} \#\ \langle\text{START}\rangle \\ \#\ \text{Awful} \\ \#\ \text{plot} \\ \#\ \text{and} \\ \#\ \text{actors.} \\ \#\ \text{The} \\ \#\ \text{movie} \\ \#\ \text{is} \\ \#\ \text{good} \\ \#\ \text{bad} \end{array}.$$

All biases are zero. The initial RNN state is $\boldsymbol{h}_0 = [0,0]^\top$ and the activation function for the RNN is the ReLU function. What will be the prediction? Show all calculations.

Solution: We have

$$\begin{aligned}
\boldsymbol{h}_1 &= \mathrm{relu}(\boldsymbol{W}_{hx}\boldsymbol{x}_{\langle\text{START}\rangle} + \boldsymbol{W}_{hh}\boldsymbol{h}_0) = \mathrm{relu}([0,0]^\top + [0,0]^\top) = [0,0]^\top. \\
\boldsymbol{h}_2 &= \mathrm{relu}(\boldsymbol{W}_{hx}\boldsymbol{x}_{\text{Awful}} + \boldsymbol{W}_{hh}\boldsymbol{h}_1) = \mathrm{relu}([-5,5]^\top + [0,0]^\top) = [0,5]^\top. \\
\boldsymbol{h}_3 &= \mathrm{relu}(\boldsymbol{W}_{hx}\boldsymbol{x}_{\text{plot}} + \boldsymbol{W}_{hh}\boldsymbol{h}_2) = \mathrm{relu}([1,0]^\top + [0,5]^\top) = [1,5]^\top. \\
\boldsymbol{h}_4 &= \mathrm{relu}(\boldsymbol{W}_{hx}\boldsymbol{x}_{\text{and}} + \boldsymbol{W}_{hh}\boldsymbol{h}_3) = \mathrm{relu}([0,0]^\top + [1,5]^\top) = [1,5]^\top. \\
\boldsymbol{h}_5 &= \mathrm{relu}(\boldsymbol{W}_{hx}\boldsymbol{x}_{\text{actors.}} + \boldsymbol{W}_{hh}\boldsymbol{h}_4) = \mathrm{relu}([-1,0]^\top + [1,5]^\top) = [0,5]^\top. \\
\boldsymbol{h}_6 &= \mathrm{relu}(\boldsymbol{W}_{hx}\boldsymbol{x}_{\text{The}} + \boldsymbol{W}_{hh}\boldsymbol{h}_5) = \mathrm{relu}([0,0]^\top + [0,5]^\top) = [0,5]^\top. \\
\boldsymbol{h}_7 &= \mathrm{relu}(\boldsymbol{W}_{hx}\boldsymbol{x}_{\text{movie}} + \boldsymbol{W}_{hh}\boldsymbol{h}_6) = \mathrm{relu}([-2,0]^\top + [0,5]^\top) = [0,5]^\top. \\
\boldsymbol{h}_8 &= \mathrm{relu}(\boldsymbol{W}_{hx}\boldsymbol{x}_{\text{is}} + \boldsymbol{W}_{hh}\boldsymbol{h}_7) = \mathrm{relu}([0,0]^\top + [0,5]^\top) = [0,5]^\top.
\end{aligned}$$

The logit for "good" is $[\boldsymbol{W}_{yh}\boldsymbol{h}_8]_{\text{good}} = 0 \times 1 + 5 \times (-1) = -5$. This is less than the logit for "bad", which is $[\boldsymbol{W}_{yh}\boldsymbol{h}_8]_{\text{bad}} = 0 \times (-1) + 5 \times 1 = 5$. Therefore the system will predict a negative sentiment (-1).

2. (5 points) Show that, if the ReLU activations are replaced by linear activations, then the sentiment classifier becomes a linear classifier as a function of the word embeddings. More precisely, show that in this case we obtain $\hat{y}_t = \text{sign}(a_0 + \boldsymbol{a}_1^\top \boldsymbol{x}_{(1)} + \ldots + \boldsymbol{a}_t^\top \boldsymbol{x}_{(t)})$ for bias a_0 and vectors $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_t$ which depend only on the RNN parameters, and where $\boldsymbol{x}_{(j)}$ denotes the word at position j.

Solution: With linear activations, we have $\boldsymbol{h}_j = \boldsymbol{W}_{hx}\boldsymbol{x}_{(j)} + \boldsymbol{W}_{hh}\boldsymbol{h}_{j-1}$. Applying this recursively, we obtain

$$\boldsymbol{h}_t = \boldsymbol{W}_{hh}^t \boldsymbol{h}_0 + \sum_{j=1}^{t} \boldsymbol{W}_{hh}^{t-j} \boldsymbol{W}_{hx} \boldsymbol{x}_{(j)}.$$

The decision is +1 if $[\boldsymbol{W}_{yh}\boldsymbol{h}_t]_{\text{good}} \geq [\boldsymbol{W}_{yh}\boldsymbol{h}_t]_{\text{bad}}$ and -1 otherwise. Letting $\boldsymbol{w}_{\text{good}}$ and $\boldsymbol{w}_{\text{bad}}$ be the corresponding rows of the matrix $\boldsymbol{W}_{yh}$, this is equivalent to

$$\begin{aligned}\hat{y} &= \text{sign}((\boldsymbol{w}_{\text{good}} - \boldsymbol{w}_{\text{bad}})^\top \boldsymbol{h}_t) \\ &= \text{sign}\left((\boldsymbol{w}_{\text{good}} - \boldsymbol{w}_{\text{bad}})^\top \left(\boldsymbol{W}_{hh}^t \boldsymbol{h}_0 + \sum_{j=1}^{t} \boldsymbol{W}_{hh}^{t-j} \boldsymbol{W}_{hx} \boldsymbol{x}_{(j)}\right)\right),\end{aligned}$$

which has the desired form with $a_0 = (\boldsymbol{w}_{\text{good}} - \boldsymbol{w}_{\text{bad}})^\top \boldsymbol{W}_{hh}^t \boldsymbol{h}_0$ and $\boldsymbol{a}_j^\top = (\boldsymbol{w}_{\text{good}} - \boldsymbol{w}_{\text{bad}})^\top \boldsymbol{W}_{hh}^{t-j} \boldsymbol{W}_{hx}$ for $j = 1, \ldots, t$.

3. (10 points) Now suppose we use a (very simple) transformer (in fact, only a single self-attention layer with a single attention head; no feedforward layers and no residual connections) as the language model. The word embeddings are the same as before, the positional embeddings are

$$\boldsymbol{p}_1 = [0, 0, 0]^\top, \quad \boldsymbol{p}_2 = [0, 0, 1]^\top, \quad \boldsymbol{p}_3 = [0, 0, 2]^\top, \quad \ldots, \quad \boldsymbol{p}_t = [0, 0, t-1]^\top.$$

and the projection matrices are

$$\boldsymbol{W}_Q = \boldsymbol{W}_K = \boldsymbol{W}_V = \begin{bmatrix} 0.1 & 0 \\ 0 & -0.1 \\ 0 & 0.1 \end{bmatrix}.$$

Scaled dot product attention is used to compute the attention probabilities. The linear output layer uses the same matrix $\boldsymbol{W}_{yh}$ as above. What will be the prediction? Show all calculations. (Hint: notice that you do not need to compute the full attention probability matrix, only the attention probabilities associated to the last word.)

Solution: By summing the word and positional encodings, we obtain the following embedding matrix:

$$\boldsymbol{X} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -5 & 1 \\ 1 & 0 & 2 \\ 0 & 0 & 3 \\ -1 & 0 & 4 \\ 0 & 0 & 5 \\ -2 & 0 & 6 \\ 0 & 0 & 7 \end{bmatrix}$$

The query, key, and value matrices are:

$$\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}_Q = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -5 & 1 \\ 1 & 0 & 2 \\ 0 & 0 & 3 \\ -1 & 0 & 4 \\ 0 & 0 & 5 \\ -2 & 0 & 6 \\ 0 & 0 & 7 \end{bmatrix} \begin{bmatrix} 0.1 & 0 \\ 0 & -0.1 \\ 0 & 0.1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0.6 \\ 0.1 & 0.2 \\ 0 & 0.3 \\ -0.1 & 0.4 \\ 0 & 0.5 \\ -0.2 & 0.6 \\ 0 & 0.7 \end{bmatrix}.$$

Since all projection matrices are the same, we have $\boldsymbol{Q} = \boldsymbol{K} = \boldsymbol{V}$. For computing the attention probabilities for the last token, we only need the last row of $\boldsymbol{Q}$, $\boldsymbol{q}_8^\top = [0, 0.7]$ and $\boldsymbol{K}$. The attention scores for this last token are

$$\boldsymbol{s}_8^\top = \boldsymbol{q}_8^\top \boldsymbol{K}^\top = \frac{1}{\sqrt{2}}[0, 0.7] \begin{bmatrix} 0 & 0 \\ 0 & 0.6 \\ 0.1 & 0.2 \\ 0 & 0.3 \\ -0.1 & 0.4 \\ 0 & 0.5 \\ -0.2 & 0.6 \\ 0 & 0.7 \end{bmatrix}^\top = \frac{1}{\sqrt{2}}[0, 0.42, 0.14, 0.21, 0.28, 0.35, 0.42, 0.49].$$

The attention probability vector is

$$\boldsymbol{p}_8^\top = [0.101, 0.136, 0.112, 0.118, 0.124, 0.130, 0.136, 0.143].$$

The representation for the last token after the self-attention layer is

$$\boldsymbol{z}_8^\top = \boldsymbol{p}_8^\top \boldsymbol{V} = [-0.028, 0.436].$$

The logit for "good" is $[\boldsymbol{W}_{yh}\boldsymbol{z}_8]_{\text{good}} = (-0.028) \times 1 + 0.436 \times (-1) = -0.464$. This is less than the logit for "bad", which is $[\boldsymbol{W}_{yh}\boldsymbol{z}_8]_{\text{bad}} = (-0.028) \times (-1) + 0.436 \times 1 = 0.464$. Therefore the system will predict a negative sentiment ("-").